UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 866

Improving Low-Power Wireless Protocols with Timing-Accurate Simulation

FREDRIK ÖSTERLIND

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2011

ISSN 1651-6214
ISBN 978-91-554-8182-7
urn:nbn:se:uu:diva-159886

Dissertation presented at Uppsala University to be publicly examined in Auditorium Minus, Gustavianum, Akademigatan 3, Uppsala, Thursday, November 24, 2011 at 13:45 for the degree of Doctor of Philosophy. The examination will be conducted in English.

Abstract
Österlind, F. 2011. Improving Low-Power Wireless Protocols with Timing-Accurate Simulation. Acta Universitatis Upsaliensis. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 866. 69 pp. Uppsala. ISBN 978-91-554-8182-7.

Low-power wireless technology enables numerous applications in areas from environmental monitoring and smart cities, to healthcare and recycling. But resource-constraints and the distributed nature of applications make low-power wireless networks difficult to develop and understand, resulting in increased development time, poor performance, software bugs, or even network failures. Network simulators offer full non-intrusive visibility and control, and are indispensible tools during development. But simulators do not always adequately represent the real world, limiting their applicability.

In this thesis I argue that high simulation timing accuracy is important when developing high-performance low-power wireless protocols. Unlike in generic wireless network simulation, timing becomes important since low-power wireless networks use extremely timing-sensitive software techniques such as radio duty-cycling. I develop the simulation environment Cooja that can simulate low-power wireless networks with high timing accuracy.

Using timing-accurate simulation, I design and develop a set of new low-power wireless protocols that improve on throughput, latency, and energy-efficiency. The problems that motivate these protocols were revealed by timing-accurate simulation. Timing-accurate software execution exposed performance bottlenecks that I address with a new communication primitive called Conditional Immediate Transmission (CIT). I show that CIT can improve on throughput in bulk transfer scenarios, and lower latency in many-to-one convergecast networks. Timing-accurate communication exposed that the hidden terminal problem is aggravated in duty-cycled networks that experience traffic bursts. I propose the Strawman mechanism that makes a radio duty-cycled network robust against traffic bursts by efficiently coping with hidden terminals.

The Cooja simulation environment is available for use by others and is the default simulator in the Contiki operating system since 2006.

Keywords: Low-Power Wireless Protocols, Wireless Sensor Networks, Contiki, Cooja, Simulation

Fredrik Österlind, Uppsala University, Department of Information Technology, Box 337, SE-751 05 Uppsala, Sweden.

ISSN 1651-6214
ISBN 978-91-554-8182-7
urn:nbn:se:uu:diva-159886 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-159886)

Swedish Institute of Computer Science
Doctoral Thesis
SICS Dissertation Series 56
ISSN 1101-1335
ISRN SICS-D--56--SE

Improving Low-Power Wireless Protocols with Timing-Accurate Simulation

Fredrik Österlind

November 2011

Swedish Institute of Computer Science
Stockholm, Sweden

To My Love, Family, and Friends

Acknowledgements

I first thank my thesis advisors Thiemo Voigt and Adam Dunkels for being my excellent advisors, and also for being the great colleagues you still are. Thiemo and Adam have complemented each other superbly, giving me feedback, perspectives, insights, and having made every single new venture truly fun.

My deepest thanks also go to my professor Per Gunningberg, Uppsala University. Per has in a seemingly easy way spotted erroneous assumptions, has valued my research contributions, and has connected my own work with another broader world of research.

I want to thank all colleagues in the Networked Embedded Systems group: Joakim, Joel, Luca, Marcus, Niclas, Nicolas, Niklas, Pablo, Shahid, Simon, Zhitao, and of course Adam and Thiemo. We have done some amazing work since the group was formed 6 years ago. Thank you all for a productive atmosphere and for unforgettable memories of us working towards 7am paper deadlines.

I am also most grateful to Sverker Janson, head of the Computer Systems Laboratory and a very supportive boss. I thank all colleagues at SICS in particular Christer Norström, Janusz Launberg, Karin Fohlstedt, Eva Gudmundsson, Lotta Jörsäter, Kersti Hedman, Orc Lönn, Thomas Ringström, Vicki Knopf, Olle Olsson, Lars Rasmusson, Bengt Ahlgren, Björn Grönvall, Martin Nilsson, Lalle Albertsson, Laura Feeney, Ian Marsh, Jarmo Laaksolahti, and Javier Ubillos.

I've had the pleasure of collaborating with a few other research groups over the years. I want to thank Mikael Johansson, Haibo Zhang, Pablo Soldati, Euhanna Ghadimi, Olaf Landsiedel, and Carlo Fischione from the Royal Institute of Technology. For past and future projects, I also thank Raimondas Sasnauskas and Klaus Wehrle from ComSys, RWTH Aachen, and Jonas Neander, Mikael Gidlund, and Igor Konovalov from ABB.

I was honored with a summer internship at Microsoft Research Redmond in 2007. I want to thank all the nice people I met while at MSR, in particular my internship mentor Bodhi Priyantha, Feng Zhao who invited me, Jie Liu, Michel Goraczko, Amal Kansal, Nupur Kothari, Mike Liang, and finally Leonardo B. Oliviera. I really enjoyed the summer in Seattle. I also want to thank Richard Han, University of Colorado, and Wen Hu and Tim Wark at CSIRO, Australia, for their hospitality.

I further thank all Contiki and Cooja developers, for providing timely and insightful replies to all and any questions. Regarding Cooja, I especially want to thank Joakim Eriksson, author of MSPsim, and Niclas Finne for numerous discussions and a pool of infinite knowledge.

Above all, I want to thank my family. My parents Kerstin and Anders, my brother Jonas, and my sister Susanne. And of course Marlene, who knows me much better than anyone else. You make me a better person.

Fredrik Österlind
Stockholm, October 2011

Throughout the work on the thesis my research has been funded by a number of projects financed by SSF (PROMOS), VINNOVA, ITEA, the Swedish Energy Agency, Stiftelsen för Internetinfrastruktur, SICS Center for Networked Systems (CNS), and Uppsala VINN Excellence Center for Wireless Sensor Networks (WISENET). My work has also been supported by the European Commission with contract FP7-2007-2-224053 (CONET). The final write-up of the thesis has been partially funded by SSF's ProInstitute Grant.

The Swedish Institute of Computer Science is sponsored by TeliaSonera, Ericsson, Saab AB Business Area Security and Defence Solutions, FMV (Swedish Defence Materiel Administration), Green Cargo (Swedish freight railway operator), ABB, and Bombardier Transportation.

Summary in Swedish

Energisnåla trådlösa nätverk har en stor mängd användningsområden, däribland miljö- och industriövervakning, smarta hem och städer, hälsa och sjukvård. Energisnåla trådlösa nätverk består av många resursbegränsade små datorer som kommunicerar via radio. Dessa datorer har lite minne, svag processorkraft, kort radioräckvidd och ett väldigt begränsat användargränssnitt. Det är svårt att utveckla mjukvara till energisnåla trådlösa nätverk, vilket leder till långa utvecklingstider, dålig prestanda, mjukvarubuggar och i värsta fall nätverk som inte fungerar.

Nätverkssimulatorer används ofta för att underlätta utveckling av energisnåla trådlösa nätverk. Simulatorer tillhandahåller en miljö där utvecklaren kan inspektera och kontrollera nätverket till fullo. Men simulationer baseras på modeller och skillnader mellan simulationsmodeller och verkliga nätverk kan begränsa eller försvåra simulationsbaserad utveckling.

I denna avhandling argumenterar jag för detaljerad modellering av tid då simulationsmiljöer används för att utveckla energisnåla trådlösa kommunikationsprotokoll. Detaljerad modellering av tid är viktigt då dessa energisnåla kommunikationsprotokoll ofta bygger på extremt tidskänsliga tekniker för att nå hög prestanda. En vanlig sådan energibesparande teknik är att driftcykla radion, så att radion är avstängd när den inte behövs. Då resurserna är knappa kan väldigt små implementations- och konfigurationsändringar påverka ett driftcyklingsprotokolls prestanda avsevärt. Under avhandlingsarbetet har jag utvecklat simulationsmiljön Cooja. Cooja kan simulera nätverk med hög detaljnivå gällande tid när Cooja kopplas ihop med emulatorn MSPsim.

Med hjälp av tidsdetaljerad simulation i Cooja utvecklar jag nya energisnåla trådlösa kommunikationsprotokoll som har hög överföringshastighet och låg latens. De två huvudsakliga insikterna som motiverar dessa protokoll gjordes i simulationsbaserade experiment. Dessa insikter är att datapaketskopiering på resurssnåla datorer utgör en flaskhals i höghastighetsprotokoll, samt att driftcykling av radion ökar risken för radiokollisioner i nätverk där alla radiosändare inte når varandra.

Jag föreslår ett nytt kommunikationsprimitiv, Conditional Immediate Transmission, med vilket flaskhalsen datapaketskopiering kan undvikas. Jag har med Conditional Immediate Transmission byggt kommunikationsprotokoll med avsevärt högre överföringshastighet än tidigare protokoll. Andra forskare har även använt tekniken för att förbättra prestandan i deras protokoll.

För att undvika problemen med radiokollisioner i driftcyklade nätverk föreslår jag kommunikationsprotokollet Strawman. Strawman hanterar radiokollisioner i driftcyklade nätverk effektivt. Istället för att undvika radiokollisioner så fördelar Strawman bandbredden mellan de kolliderande radiosändarna direkt när den upptäcker en radiokollision. Till skillnad från tidigare protokoll så kan Strawman effektivt schemalägga radiosändare som inte hör varandra, och löser därmed det så kallade hidden terminal-problemet.

Cooja är fritt tillgänglig för forskare och utvecklare sedan år 2006 och är standardsimulatorn i operativsystemet Contiki.

List of Papers

This thesis is based on the following papers.

Paper A F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Tampa, Florida, USA, November 2006

Paper B F. Österlind and A. Dunkels. Approaching the Maximum 802.15.4 Multihop Throughput. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, Charlottesville, Virginia, USA, June 2008

Paper C H. Zhang, F. Österlind, P. Soldati, T. Voigt, and M. Johansson. Rapid Convergecast on Commodity Hardware: Performance Limits and Optimal Policies. In *The IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON)*, Boston, Massachusetts, USA, June 2010

Paper D F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet Checkpointing: Enabling Repeatability in Testbeds and Realism in Simulations. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Cork, Ireland, February 2009

Paper E F. Österlind, N. Wirström, N. Tsiftes, N. Finne, T. Voigt, and A. Dunkels. StrawMAN: Making Sudden Traffic Surges Graceful in Low-Power Wireless Networks. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, Killarney, Ireland, June 2010

Paper F F. Österlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels. Strawman: Resolving Collisions Through Collisions. In submission

Publications Not In Thesis

- E. Ghadimi, P. Soldati, F. Österlind, H. Zhang, and M. Johansson. Hidden terminal-aware contention resolution with an optimal distribution. In *The Eighth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2011
- S. Duquennoy, F. Österlind, and A. Dunkels. Lossy Links, Low Power, High Throughput. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, November 2011
- I. Konovalov, J. Neander, M. Gidlund, F. Österlind, and T. Voigt. Evaluation of WirelessHART Enabled Devices in a Controlled Simulation Environment. In *IEEE International Symposium on Industrial Electronics*, Gdansk, Poland, 2011
- A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne. The announcement layer: Beacon coordination for the sensornet stack. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Bonn, Germany, 2011
- C. A. Boano, K. Römer, F. Österlind, and T. Voigt. Demo Abstract: Realistic Simulation of Radio Interference in COOJA. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Bonn, Germany, 2011
- F. Österlind, J. Eriksson, and A. Dunkels. Demo Abstract: Cooja TimeLine: A Power Visualizer for Sensor Network Simulation. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, 2010
- F. Österlind, R. Sasnauskas, O. S. Dustmann, A. Dunkels, and K. Wehrle. Demo Abstract: Integrating Symbolic Execution with Sensornet Simulation for Efficient Bug Finding. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, 2010
- Q. Li, F. Österlind, T. Voigt, S. Fischer, and D. Pfisterer. Making Wireless Sensor Network Simulators Cooperate. In *Seventh ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, Bodrum, Turkey, 2010
- N. Tsiftes, J. Eriksson, N. Finne, F. Österlind, J. Höglund, and A. Dunkels. A Framework for Low-Power IPv6 Routing Simulation, Experimentation,

and Evaluation. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM), demo session*, New Delhi, India, August 2010

- A. Dunkels, F. Österlind, and N. Tsiftes. IP-based Sensor Networks: A Hands-on Tutorial. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, San Francisco, California, USA, 2009
- J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, and T. Voigt. Accurate Network-Scale Power Profiling for Sensor Network Simulators. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Cork, Ireland, February 2009
- J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. Mspsim – an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Delft, The Netherlands, January 2007
- F. Österlind, E. Pramsten, D. Roberthson, J. Eriksson, N. Finne, and T. Voigt. Integrating Building Automation Systems and Wireless Sensor Networks. In *12th IEEE Conference on Emerging Technologies and Factory Automation*, Patras, Greece, 2007
- A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, November 2007
- A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, Cork, Ireland, June 2007

Contents

Part I:
Thesis Summary

1. Introduction

Low-power wireless technology is envisioned to have a greater impact on society than the Internet. Low-power wireless is an enabling technology for machine-to-machine networks, sensor and actuator networks, and the Internet of Things (IoT). The telecommunications company Ericsson estimates that more than 50 billion low-power wireless devices will be deployed and connected to Internet by year 2020 [37] with applications ranging from smart cities and the smart grid, to healthcare, recycling, and food traceability [39, 122].

Low-power wireless networks consist of extremely resource-constrained devices. A low-power device typically has a few tens of kilobytes of memory, a processor running at a few megahertz, a limited power budget, and less than 100 meters radio communication range. Furthermore, the devices are embedded and low-cost as each device must operate autonomously for years, and cost no more than a few Euros.

The resource constraints of low-power devices make development of low-power wireless systems tedious and error-prone. Since the devices are often battery-powered, they must preserve energy to maximize network lifetime. To meet these resource limitations, specialized operating systems and communication protocols have been developed. But the resource constraints limit visibility into the network execution, which hampers understanding and makes software bugs difficult to track down.

Simulation is frequently used to develop low-power wireless systems. Simulation offers full visibility and control of an entire network and allows a developer to quickly test, debug, and improve applications and protocols. However, simulators do not always adequately represent real environments, which limits their use as development tools. For example, simulators may be unable to simulate real-hardware applications, i.e., applications that can be programmed to real-hardware devices. The developers must consequently translate simulated applications before programming their devices, potentially changing their behavior or causing software bugs.

This thesis undertakes three research challenges. First, I consider how simulation can be used to facilitate development of low-power wireless applications, and what properties are needed in the simulation environment. Second, I address what network primitives and abstractions are needed to enable high-performance networking protocols. Third, I look into the relationship between radio duty-cycling protocols and bursty traffic patterns.

My main contribution in this thesis is demonstrating that timing-accurate simulation [Paper A, Paper D] is important when developing low-power wireless protocols with high performance. More specifically, I use the timing-accurate simulation environment Cooja to develop extremely timing-sensitive protocols, building upon already existing radio duty-cycling techniques that are central to low-power wireless networks [Paper B, Paper C, Paper E, Paper F].

I have built and experimented with numerous low-power wireless networks over the last few years. As a researcher my purpose has been to gain understanding of how these networks behave, learn their limitations, and push state-of-the-art performance.

Limited by the functionality offered by the simulators at hand, I started developing the simulation environment Cooja in 2005. Cooja can, when combined with the device emulator MSPsim [38], simulate low-power wireless networks with high timing accuracy. Furthermore, the simulated applications can without modifications be uploaded to real-hardware devices. Cooja is now an integral part of the Contiki operating system development environment, and has also been used by others to perform experiments [9, 60, 75, 89, 90, 97, 108, 110, 115, 117] .

To simplify and accelerate development, I have strived to minimize the gap between simulation and real-hardware networks. Consequently, Cooja can be configured to simulate networks using different amounts of detail [Paper A], that span over both real-hardware and simulated devices [66], and that co-exist entirely in both simulation and on real hardware [Paper D].

I have found that high timing accuracy and simulating real code are two imperative properties of a simulation environment used for developing low-power wireless networks. The need for high timing accuracy stems from the extremely timing-sensitive radio duty-cycling techniques that are used to reduce power consumption in low-power wireless networks. By contrast, wireless simulation environments that do not target low-power development have less stringent requirements on timing accuracy [54]. Simulating deployable code allows a developer to quickly iterate between simulation and hardware-based experiments.

1.1 Low-Power Wireless Networks

Low-power wireless technology is a basic building block in machine-to-machine networks, wireless sensor and actuator networks, and the Internet of Things. A typical low-power wireless network consists of battery-powered devices, each equipped with a microcontroller, a short-ranged radio, and sensors and actuators. These devices collaborate to gather information about the physical environment and to securely transfer it to a user, possibly over the Internet. The network must use scalable and robust communication

protocols, where neighboring devices help each other to extend radio communication range, and to overcome fluctuating and bad communication links. Finally, the network must be energy-efficient as batteries should provide several years of network lifetime.

1.1.1 Applications

Low-power wireless technology can be used to implement numerous different applications [39]. The smart grid is the next generation intelligent electricity network. Smart meters are an important part of the smart grid. Smart meters collect fine-grained per-house electrical usage data in real-time for use by both energy suppliers and end-users, and are already being deployed in large numbers. For example, Smart Meter Texas (SMT) is a collaboration between electricity suppliers in Texas, USA [106]. The SMT website currently provides millions of Smart Meter-equipped end-users with their electrical usage updated every 15 minutes. The smart grid enables energy suppliers to adjust prices depending on the current load.

In the medical domain, low-power wireless networks can enable longer independent living for the aging population. Wearable or implantable sensor nodes monitor and report patient vitals to the hospital regularly and upon emergencies. This technology enables rehabilitating patients to leave hospitals earlier and cuts healthcare costs [39]. The CodeBlue research project led by the Harvard Sensor Networks Lab develops low-power applications for medical care [52]. CodeBlue has developed a small wearable device that monitors a patient's heart rate, blood oxygen saturation, and EKG. The device automatically raises an alarm if monitored values fall outside a normal range.

Another example which demonstrates how low-power wireless technology will be used in future smart cities is the San Francisco-based SFPark company [80]. SFPark monitors parking spaces in San Francisco and allows drivers to see available parking spaces in their vicinity via an iPhone app. SFPark not only demonstrates how this technology can reduce pollution and save times for drivers, but also how new market opportunities arise; parking space prices can now be adjusted depending on availability in the area.

1.1.2 Power

Wide-spread development and deployment of low-power wireless networks face a number of challenges, one of the most prominent being how to preserve power and thus prolong network lifetime. These networks are equipped with radios so a wired communication infrastructure is no longer needed. This may reduce installation costs and enable temporary deployments. To fully avoid the need for cables the networks are often battery-powered. Coupling battery-powered networks with the additional requirements of small physical device

sizes and low-cost hardware components imposes severe restrictions on the hardware and software.

Power limitations have guided low-power wireless research for the last decade and have motivated the use of low-power hardware. For example, low-power microcontrollers support deep-sleep modes that consume only a tiny fraction of the power spent when awake. In addition, microcontrollers have very limited memory and processing abilities [79]. Low-power radio transceivers operate with low bit rates and have shorter transmission ranges than what is available by other devices, for example laptops [58].

The use of low-power hardware must be complemented by software techniques to further reduce power consumption. For example, hardware components must be turned off when not needed; the components are duty cycled by the on-board software. For some components, such as the microcontroller, duty cycling is easy to implement. When the operating system has no immediate tasks to execute, it schedules a microcontroller wake up timer and enters a sleep-mode. Duty cycling other hardware components can be extremely difficult and has been thoroughly researched. For example, a sensor device that monitors the physical environment cannot detect sounds or vibrations if its sensor components are turned off. Duty cycling therefore requires careful consideration if not to negatively affect the running application.

Duty cycling the radio transceiver to reduce power consumption is particularly challenging. Low-power devices in a multi-hop network collaboratively forward each other's radio messages. But the radio transceiver must be awake to detect, receive, and forward a radio message. Since low-power short-ranged radios consume roughly the same power when listening for incoming messages as when transmitting messages [20], listening for incoming messages is a major power consumer. Radio duty-cycling protocols therefore offer significant power consumption savings by reducing the amount of idle listening. Numerous different techniques have been proposed as discussed in Section 1.2.

Low-power wireless communication protocols must cope with the inherent challenges of radio environments, such as unstable and bursty links [109], asymmetric links [46], and hidden terminals. The potato field sensor network deployment by TU Delft [71] demonstrates how unstable radio links can lead to data collection losses. In particular, the 100-node agriculture deployment experienced data losses due to fluctuating radio links that were affected by differences in day and night-time humidity [116].

Asymmetric links are characterized by one-way communication; a node is able to receive from but not send to a neighbor. Asymmetric links are commonly found in short-ranged wireless networks [46]. The hidden terminal problem occurs when two devices that are out of radio range send to a common neighbor at the same time. Since they cannot detect each other's transmissions, they may both transmit at the same time causing radio collisions and data loss at the common neighbor. Low-power wireless protocols should

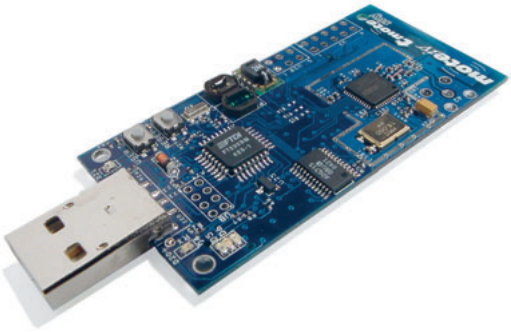

Figure 1.1: The Tmote Sky is a popular low-power wireless device in research projects. It is equipped with an MSP430 microcontroller and a short-ranged low-power CC2420 radio transceiver.

be designed to cope with challenges like unstable and asymmetric links and hidden terminals.

1.1.3 Hardware

A low-power wireless device consists of a microcontroller unit (MCU), a wireless radio transceiver, a set of sensors or actuators, and a power source such as a battery. The resources are typically very scarce, both concerning processing, memory, and communication bandwidth. A typical MCU has an 8 or 16-bit memory architecture, with 20-50 kB programmable ROM, 2-10 kB RAM, and is running at a few megahertz—several orders of magnitude less than an ordinary laptop. Even though there are more powerful MCUs, they have not seen widespread use due to their higher power consumption and cost.

A popular device used in many research projects and in this thesis is the Tmote Sky developed by MoteIV [79, 94]. The Tmote Sky, shown in Figure 1.1, employs a TI MSP430F1611 microcontroller with 48 kB ROM, 10 kB RAM, and a processor clock frequency of up to 8 MHz. The Tmote Sky is equipped with an IEEE 802.15.4-compliant TI CC2420 radio transceiver [20]. The CC2420 is a short-ranged low-power radio with a communication range of less than 100 meters and a bitrate of 250kbit/s.

The Tmote Sky has on-board sensors to measure temperature, humidity, and light. Other low-power devices provide different sensors and research projects have collected temperature, humidity, power consumption, air or noise pollution, volcano eruptions, heart rates, and workout efficiency.

1.1.4 The Contiki Operating System

To meet the resource constraints of low-power devices, operating systems such as Contiki [27], TinyOS [55], SOS [50], and Mantis [11] have been developed. Contiki is an open-source operating system specifically targeting networked embedded systems and the Internet of Things [27]. The Contiki development has been lead by Adam Dunkels since its first release in year 2003. Contiki has received significant development contributions from industry including Cisco, Atmel, and SAP. Contiki runs on many of the available low-power MCUs, radios, and low-power platforms.

Contiki provides three network stacks: μIP [28], μIPv6 [21], and Rime [30]. μIP is an IPv4 stack for memory-constrained devices [28]. The IPv6 stack μIPv6 was developed by Cisco as an add-on to μIP, and was contributed back to the open-source community via Contiki [21]. Contiki has an implementation of the IPv6 routing protocol RPL, to support low-power multi-hop routing over lossy networks [65].

The Rime networking stack, like μIPv6, enables low-power and low-resource multi-hop networking. In contrast to IPv6, Rime is tailored specifically for low-power wireless networks and can be used as a performance baseline in comparisons. Rime consists of a set of thin network layers and a few transformation modules that allow Rime-generated network traffic to simulate other network stacks [30]. Both μIP, μIPv6, and Rime can run on top of Contiki's radio duty-cycling protocols.

Contiki provides a software-based energy estimation module that measures the individual power consumption of the different on-board components. The energy estimation module attributes power consumption to activities such as microcontroller sleep mode, radio listen, and radio transmit [31].

Contiki also comes with the Cooja/MSPsim simulation environment that enables deployable Contiki code to be simulated and tested before deployment. I am the main developer of Cooja and much of the work put into developing Contiki's simulation environment has been done as a part of this thesis. Cooja/MSPsim consists of the two simulators Cooja and MSPsim. Cooja is a modular network simulator that allows for networks to be simulated in different radio environments. Cooja supports cross-level simulation that simulates different network nodes at different detail levels. MSPsim is an emulator for the Texas Instruments' MSP430 microcontroller family and has been integrated with Cooja to allow for network emulation of Tmote Sky devices.

1.2 Achieving Low-Power Operation

Preserving energy to prolong network lifetime is one of the main research challenges in low-power wireless technology. This affects the design of the entire system, ranging from applications and networking protocols, to the individual hardware components. To avoid spending unnecessary energy, power-

consuming hardware components are regularly turned off when not needed; the microcontroller, radio, and sensors are duty cycled.

1.2.1 Radio Duty-Cycling Challenges

The radio is one of the most important and difficult hardware components to duty cycle. Radio communication requires the radio to be turned on, but the radio is one of the most power-expensive hardware components in low-power wireless devices. Short-ranged low-power radios, such as the IEEE 802.15.4 CC2420 radio transceiver [20], consume roughly the same power when listening for incoming radio transmissions as when transmitting. For example, the Tmote Sky consumes 21.8 mA in radio listen mode and 19.5 mA in radio transmission mode [79]. This is an artifact of the low transmission power that low-power radios use. As a comparison, WiFi radios typically transmit at 15-20 dBm (32-100 mW), whereas the CC2420's maximum transmission power is 0 dBm (1 mW) [20]. By contrast, WiFi radios offer a significantly higher transmission bitrate than low-power radios. For example, IEEE 802.11g runs at 54 Mbits/s compared to the IEEE 802.15.4 bitrate at 250 kbit/s.

Radio duty-cycling protocols strive to reduce power consumption by turning the radio off when not needed. Idle listening—keeping the radio on when no data is being transmitted or received—often consumes the majority of the energy spent in a network. Several duty-cycling techniques and protocols have been proposed to reduce the amount of idle listening, and experimental deployments have reported duty cycles of 1-2% [120]

Radio duty-cycling protocols are commonly classified as synchronized [43, 92] or contention-based [15]. Synchronized protocols maintain a time schedule of when to receive from and transmit to neighboring devices, and keep the radio off elsewhere. Contention-based protocols do not rely on time synchronization, but instead wake up regularly and scan for incoming transmissions from neighbors.

Modern radio duty-cycling protocols often combine both synchronized and contention-based techniques. Synchronized protocols rely on accurately keeping track of neighbors' wake up schedules, but low-power devices suffer from clock drifts, which require schedules to be resynchronized [33]. Duty-cycling protocols must also cope with unstable and varying radio links, limited memory for storing neighbor schedules, and sudden devices failures or reboots. Therefore a mixed approach is often employed where synchronization is used to improve power savings, but the protocol falls back on contention-based techniques if synchronization fails [36].

A major challenge in duty-cycled networks is matching the wake up rate to the anticipated network traffic. Duty-cycling preserves energy by turning the radio off at the cost of reducing the effective network throughput and increasing the network latency. But the amount of traffic in a network is very difficult to predict pre-deployment and may furthermore vary during the net-

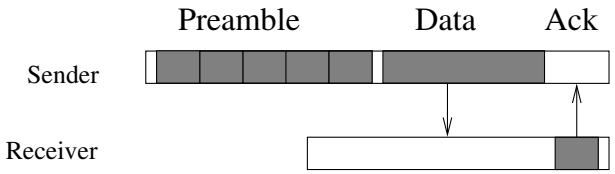

Figure 1.2: In sender-initiated LPL-based protocols, all devices wake up periodically and listen for ongoing transmissions. A sender first transmits an elongated preamble to ensure that the intended receiver has time to wake up and detect the transmission.

work lifetime, especially in event-driven networks that monitor the physical environment and only generate data upon detecting an event [6].

Radio duty-cycling techniques aggravate the risk of radio collisions and network congestion. For instance, with a contention-based protocol a single network-layer transmission may result in tens or hundreds of physical-layer transmissions. Multiple simultaneous network-layer transmissions among neighboring devices therefore have a higher risk of colliding in duty-cycled networks than in non-duty-cycled ones. Upper-layer protocols should be able to distinguish between data loss due to packet storms, unstable radio links, and malfunctioning neighbor devices.

A sender in a radio duty-cycled network must wait for neighbor's radio to wake up before data can be sent to that neighbor. An alternative approach is to use a separate ultra-low-power wake-up radio by which the sender informs the destination to immediately turn on its ordinary radio [49]. Adding a separate wake-up radio may render duty-cycling protocols unnecessary, but increases the hardware cost of each device. How to best use wake-up radios is still an active research topic and is not addressed in this thesis.

1.2.2 Sender-Initiated Duty-Cycling

Many radio duty-cycling protocols are based on a technique called low-power listening (LPL) [15, 36, 93]. In LPL-based protocols, all devices keep their radios off the majority of the time, but wake up regularly to scan for ongoing transmissions. The devices may for example be configured to wake up once per second. Since they are unsynchronized, a device about to send to one of its neighbors does not know when that neighbor will wake up—only that the neighbor will wake up briefly within the next second. Senders therefore use long transmission preambles prior to transmitting the actual data payload. When the neighbor wakes up and detects the preamble, it stays awake until the data has been received, see Figure 1.2.

Several optimizations have been proposed on top of the basic LPL-technique. Buettner et al. [15] proposed to include the destination address in the preamble, thereby avoiding the need to always send a

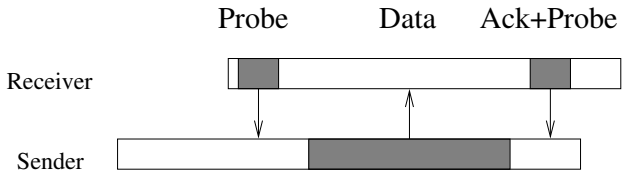

Figure 1.3: Receiver-initiated radio duty-cycling with Low-Power Probing. The receiver initiates a data transfer by sending a small data probe packet that the sender immediately responds to with the data packet transmission.

full-period preamble; when the destination node wakes up and detects the preamble, it immediately informs the sender that it is now awake. The sender can now send the data immediately, without the rest of the preamble. Surrounding non-destination neighbors, on the other hand, learn that the ongoing transmission is for someone else and turn off the radio again.

Another optimization was proposed in the WiseMAC protocol [36]. WiseMAC assumes a periodic wake up schedule and maintains a schedule of all past neighbor wake up times. This wake up optimization enables WiseMAC to delay the preamble until just before the destination neighbor wakes up, resulting in both significantly less network congestion as well as reduced power consumption.

1.2.3 Receiver-Initiated Duty-Cycling

Like in sender-initiated duty-cycling, receiver-initiated duty-cycling [35, 81, 112] allows network devices to keep their radios turned off most of the time. Low-Power Probing (LPP) is a receiver-initiated radio duty-cycling technique. In LPP-based networks the receiver initiates a data transmission. All devices regularly transmit data probes—tiny radio packets with a source address. A node that wants to send a packet just turns on its radio and waits for a data probe from the respective destination neighbor, see Figure 1.3. LPP exploits the fact that transmitting data costs roughly the same as receiving data in terms of power consumption. However, LPP improves on network congestion as it avoids transmitting the long preambles of LPL and X-MAC.

1.2.4 Implications of Radio Duty-Cycling

Radio duty-cycling can reduce the power consumption of a network significantly, but has implications that fundamentally change how the network behaves. It is therefore important to evaluate upper-layer protocols, e.g., routing and neighbor discovery together with duty-cycling protocols, or else risk drawing wrong conclusions from the experiments.

An implication of contention-based radio duty-cycling protocols is that broadcast transmissions become significantly more expensive than unicast transmissions in terms of energy and latency. Modern radio duty-cycling protocols make use of both the unicast-optimization by Buettner et al. [15] and the wakeup-optimization by El-Hoiydi et al. [36]. These optimizations reduce the overhead of unicast transmissions. But they do not reduce the overhead of broadcast transmissions since a broadcast transmission must wake up all neighbors. Consequently, broadcast transmissions become more power expensive unicast transmissions [77].

Another implication of contention-based duty-cycling protocols is that broadcast transmissions are no longer atomic operations. A single network-layer broadcast will be received by neighbors at different times, since they are unsynchronized and receive the different physical-layer transmissions at different times.

Finally, radio duty-cycling also removes the possibility of over-hearing packets for other neighbors. Since radios are mostly turned off, a device no longer overhears packets intended for its neighbor. Yet, overhearing is sometimes assumed by low-power network protocols [64].

Implications of radio duty-cycling, such as expensive and non-atomic broadcasts, and lack of overhearing, may affect the performance of upper-layer protocols. These upper-layer protocols, if they are to be used in radio duty-cycled networks, should therefore be evaluated together with radio duty-cycling protocols.

1.3 Developing Low-Power Protocols

Low-power wireless protocols provide connectivity to resource-scarce devices over volatile radio links while minimizing power-consumption. Low-power protocols use techniques such as radio duty-cycling to reduce power consumption, header compression and data aggregation to reduce transmissions, and energy-aware routing to balance network resources.

Development of low-power protocols includes design, prototyping, implementation, testing, experimentation, and debugging. In the design phase, the developer gathers information about the environment in which the protocol should be used. Prototyping and implementation require access to low-power devices, either simplified simulation models or real-hardware devices, for which the developer implements the protocol. In testing and experimentation phases, the developer verifies protocol correctness and measures performance. Debugging often involves iterating between different phases; a problem discovered during testing may require the developer to redesign the protocol.

Development of low-power wireless applications can be done directly against hardware devices, i.e. the program is compiled and uploaded to

low-power devices where it is then tested. Developing directly against hardware is, however, difficult. The devices are resource-constrained, which limits the visibility into the software execution. For example, the user interfaces of research prototype devices are typically limited to a few LEDs and a user button. Similarly, controlling the execution of a real-hardware network is difficult, such as injecting controlled packet loss to trigger a certain behavior. The experiments may furthermore involve a large amount of devices, which requires reprogramming and monitoring an entire network of devices. Finally, many applications should operate for a long time, so thorough testing on real hardware can be very time-consuming.

Low-power wireless protocols must cope with device clock drifts, yet the protocol mechanisms are often extremely timing-sensitive. Low-power protocols must be robust against unstable radio links, microcontroller clock drifts, and device failures. But techniques such as radio duty-cycling are timing-sensitive and must be calibrated for the low-power devices to achieve high performance. For example, a radio duty-cycling protocol may wake up the radio less than a millisecond before it anticipates an incoming transmission, and immediately turn the radio off again if no transmission is detected.

1.4 Simulation-Assisted Development

Simulation is a widely used tool for developing low-power wireless networks. Simulation offers an environment with full control and non-intrusive visibility into the network execution. It is used to perform controlled experiments and comparisons, as well as to support development phases such as design, implementation, testing, and debugging.

In this thesis, I argue that timing-accurate simulation is important when developing low-power wireless protocols, as it simplifies developing protocols with high-performance. Hardware emulators can be used to provide this high level of timing accuracy [38, 119]. An additional benefit of emulators is that they simulate the same code as is uploaded to real-hardware devices. Simulation of real-hardware code simplifies migrating from simulation to hardware, and allows the developer to rapidly iterate between simulation-based and real-hardware experiments.

Accurate simulation models reduce the risk of drawing wrong conclusions from simulation-based experiments, and many other factors besides timing-accuracy are important. Simulation models for, e.g., radio environments [136], execution timing [38, 119], power consumption [104], lifetime estimation [40], and sensed environments [53] have been developed.

Simulation-based studies of network power consumption require an accurate device-level power consumption model. Radio duty-cycling protocols may significantly reduce the power consumption, but are sensitive to timing.

Studying the power consumption in a duty-cycled network therefore requires timing-accurate simulation.

Radio environments are notoriously difficult to model, as low-power wireless networks often are deployed indoors and in the vicinity of interference from WiFi networks [74]. In particular, both IEEE 802.15.4 and WiFi (IEEE 802.11) operate on 2.4 GHz. In addition, low-power networks are affected by attenuating building materials such as concrete walls, multi-path self-interference, and even home appliances such as microwave ovens [12]. These aspects make radio environment modeling very challenging; even with an extremely detailed simulation model, that model does not necessarily represent the intended deployment location. The same reasoning is also true for hardware testbeds since the radio behavior in a hardware testbed does not necessarily represent that of a deployment.

For some studies, the sensed environment is also important to model accurately. Event-driven low-power networks that monitor the physical environment and react to physical events need accurate models of the sensed environment, for example how temperature changes spread through a building. These networks are idle until an event occurs, and then generate large amounts of network traffic [6]. Similarly, networks that compress and aggregate sensed data require the simulated sensed data to resemble real sensed data.

The above discussion exemplifies how simulation-based experiments rely on accurate models, but the required accuracy depends on the type of simulated application. For example, consider an experiment that measures the network congestion in a temperature-monitoring application. Unless the application acts upon the observed temperature, this network will be equally congested in both cold and warm environments. As such, a simulation-based experiment does not require a detailed temperature model. Rather, unnecessarily accurate models hamper understanding and result in slower simulation execution [54]. In addition, better accuracy does not always foster easier understanding of the network behavior, and a developer may prefer trading accuracy for easier understanding in early development phases.

Simulation scalability is important as many networks are envisioned to contain thousands of devices. Simulation scalability has therefore been thoroughly addressed by the research community [73, 119, 121]. Simulation accuracy is often traded for scalability, so the developer should be able to choose the level of details with which the network is simulated. Another benefit of such flexibility is that a developer can trade accuracy for easier understanding of the network behavior. For example, an extremely simple radio environment may be preferred over an accurate environment when debugging complex interactions among neighboring nodes.

Simulation-assisted development also benefits from visualization and control tools. Visualization tools may provide the developer with a graphical representation of the current network behavior; see Figure 1.4. Simulation control tools enable the developer to pause, slow down, or restart simulations.

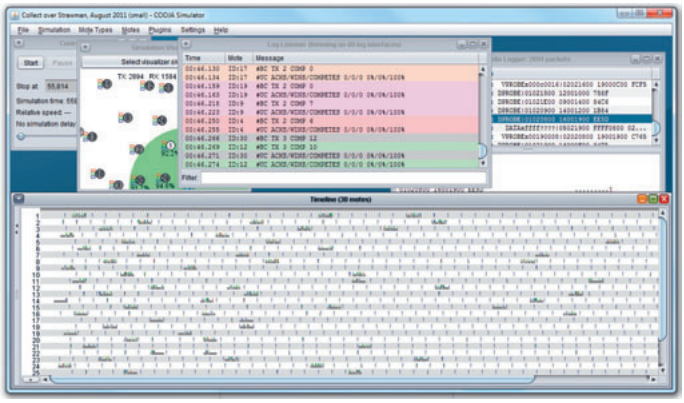

Figure 1.4: Cooja's Timeline visualizes network radio traffic and makes it possible to see the behavior of low-power radio duty-cycling protocols [86].

In conclusion, no single simulation environment fits all, and the required simulation accuracy depends heavily of what conclusions to draw from the experiments. In this thesis, I argue that high timing accuracy is important for developing high-performance low-power wireless protocols and present the timing-accurate simulation environment Cooja.

1.5 Research Methodology

The research method used throughout this thesis is to formulate a hypothesis from a research idea, and then develop a computer system that embodies the idea. The computer system is used to support experiments that test the hypothesis.

For example, the research idea in Paper A was cross-level simulation that proposes mixing low-power devices modeled at different detail levels in the same simulated network. Our hypothesis was that cross-level simulation improves simulation scalability. To test this hypothesis, I built the Cooja simulator as well as a set of specialized programs used during the experiments. A single idea may generate multiple hypotheses, and similarly, a single hypothesis may require building multiple computer systems.

My time spent on implementing and building systems has provided important insights and triggered new research ideas. As an additional bonus, the systems built may prove useful also after the idea has been evaluated. For example, the Cooja simulator was originally built to evaluate the cross-level simulation approach, but has been maintained and further developed as an environment to support development of new low-power wireless networks.

The tools I have used to build computer systems are most notably the Contiki operating system and the Cooja simulator. In addition, I have also used publicly available sensor network testbeds to perform experiments [51].

If the experiments falsify the hypothesis, I can change the hypothesis, rebuild the system, refine the experiments, or drop the idea altogether; there is always a plethora of new research ideas to pursue.

1.6 Thesis Structure

This dissertation consists of two parts. The first part is the thesis summary, and the second part contains the included publications.

Chapter 2 motivates and discusses the research challenges addressed in the thesis. Chapter 3 summarizes the research contributions of the included publications, as well as other contributions that emerged from the thesis work. The included publications are individually summarized and discussed in Chapter 4, and related work is given in Chapter 5. Finally, the first part of the thesis is concluded in Chapter 6.

2. Research Challenges

2.1 Simulation-Assisted Development

Simulation is often employed for experiments on low-power wireless networks, but the research community is skeptical towards simulation-based experimental results. Deployment experiences have reported application failures due to bugs overseen by extensive simulation-based tests [71]. A survey in the related research field Mobile Ad-hoc Networking discusses the wide use of and the many problems associated with simulation in research [69]. Consequently, simulation-based experimental results are often not well-received by the community, and are instead replaced by hardware testbed experiments [68].

Misleading simulation-based experimental results may be due to incorrect or oversimplified simulation models. For example, a radio duty-cycling protocol may render network broadcast transmissions significantly more expensive than unicast transmissions [29]. A simulation-based experiment on the performance of upper-layer protocols that use both unicasts and broadcasts therefore requires simulation models of the duty-cycling protocol, but many simulation environments do not model the radio duty-cycling layer.

The benefits of simulation go beyond evaluation: simulators are also used for application development, debugging, and understanding. During these phases, realistic simulation models may not be required, or preferred. Rather, simple models remove unnecessary execution details, and help the developer to focus on the protocol under study. As the development phase progresses, more complex models that better reflect real environments can be introduced.

A typical development process goes from design, via simulation and implementation, to testbeds experiments and evaluation[68]. Several iterations between the different phases are often needed. Simulations expose flaws that demand application re-design, and testbed experiments expose bugs not detected earlier due to over-simplified simulation models.

A main insight in this thesis is that, to promote efficient development, a developer should be able to configure the level of simulation details. Simple simulation models are used early on in the development phase. Later, as the application matures, more detailed and complex models are introduced. Traditional simulators, however, perform simulation at a single fixed detail level. Simulators with fixed detail levels include the network simulator ns/2 [121], the operating system simulator TOSSIM [73], and the hardware emulator AvroraZ [4].

This thesis presents *cross-level simulation* and its implementation in the Cooja simulator [Paper A]. Cooja supports simulation at both the network level, the operating system level, and the hardware level. Cooja's cross-level simulation allows the developer to configure networks with nodes simulated at different detail levels, and to gradually increase the amount of details in a simulated network. Cross-level simulation therefore simplifies migrating between design, simulation, and hardware testbed. Cross-level simulation also improves scalability, as devices simulated with fewer details require less memory and execution time.

To further simplify the migration from simulation to testbed, this thesis proposes *sensornet checkpointing* [Paper D]. Sensornet checkpointing allows transferring network-wide checkpoints between simulation and testbeds. A technique that is conceptually similar to sensornet checkpointing is hybrid simulation [76, 88, 124, 125]. Hybrid simulation mixes simulated devices with real-hardware devices, to form a network that spans over both simulation and hardware. In contrast to hybrid simulation, sensornet checkpointing keeps the entire network in either simulation or testbed. When the network execution is simulated, the developer enjoys simulation benefits such as repeatability and visibility. When the network is executed in the testbed, the application is tested in a realistic environment.

Cross-level simulation and sensornet checkpointing improve scalability and realism of simulation-based experiments. My experience from using these techniques, however, is that timing-accurate simulation as provided by hardware-level simulation reduces the need for cross-level simulation and sensornet checkpointing. More specifically, timing-accurate simulation allows a developer to carefully tune the implementation of a low-power wireless protocol prior to real-hardware experiments, to improve its performance. Without timing-accurate simulation, such as when simulating at the operating system-level, protocol tuning is not feasible as experimental results differ too much between simulation-based and real-hardware experiments. The level of timing-accuracy that Cooja's hardware level provides is on individual microcontroller instructions and transmitted radio bytes. MSPsim emulates software execution on the microcontroller instruction-level. Cooja simulates communication timing using individually transmitted radio bytes. I have not observed need for more detailed timing when developing and tuning low-power wireless protocols.

2.2 Enabling High-Performance Low-Power Protocols

Upper-layer communication protocols such as routing and data collection are built upon a set of lower-layer communication primitives, e.g., transmit, receive, and listen. The lower-layer communication primitives and application programming interfaces (APIs) that are available determine both the design

and performance of the upper-layer protocols. Communication primitives and APIs have been thoroughly addressed in general purpose operating systems. In particular, no-copy socket APIs have been proposed that offer increased system throughput [2, 26].

Low-power wireless protocol research has led us to revisit the communication primitives. In batch-and-send networks, as opposed to in traditional sense-and-send networks, sensor data is sampled for extended periods of time before transported out of the network. By transmitting a large batch of data instead of several small, power-saving optimizations are enabled that rely on high data throughput [33, 64]. When we worked on Paper B, however, there was a large discrepancy between the achieved throughput of state-of-the-art bulk transfer protocols [64] (10 Kbit/s), and the raw radio bitrate (250 Kbit/s).

TMDA-based convergecast protocols [43] rely on time slot schedules and time slot sizes to achieve low-latency data collection. Convergecast refers to the process of collecting data from a large set of nodes. With smaller time slots, the convergecast latency and power-consumption can be reduced. But reducing time slot lengths puts a higher burden on the devices, since they have less time to receive or send data in each slot.

Armed with our simulation framework, we study the critical path of bulk transfer and convergecast protocols and present the *Conditional Immediate Transmission* (CIT) technique. In Paper B we identify that single-channel operation and packet copying are two major performance bottlenecks. CIT uses pipelining to remove packet copying off the critical path. We show that CIT can both increase throughput in bulk transfer protocols [Paper B], and lower latency in convergecast applications [Paper C]. The CIT technique has in addition been used by others to increase performance in a pipelined bulk transfer protocol [95], and we have used it to achieve high throughput over lossy links [32].

2.3 Radio Duty-Cycling with Bursty Traffic

In radio duty cycled networks, the radio wake up rate is configured against the anticipated network traffic; idle listening is traded for elongated preambles [15]. Neighbor wake up synchronization lessens the need for transmitting full preambles [36, 132], and enables further lowering the wake up rate while maintaining the same network data. Dutta et al. discuss the lower bounds of achievable duty cycles, and argue that with typical microcontroller clock drifts the lower bound in synchronized networks is 0.01% [33], significantly lower than state-of-the-art deployments.

This thesis considers the relationship between radio duty-cycled networks and bursty traffic patterns. A simple example demonstrates this relationship. Consider a network with a fixed traffic flow, in which a given device is sent on average 1 packet every 10 seconds from its neighbors. If the device wakes

up once every second the data-per-wakeup ratio is $1/10 = 0.1$. If the wakeup rate is decreased and the node instead wakes up once every 5 seconds, the data-per-wakeup ratio becomes $10/5 = 0.5$. A higher data-per-wakeup ratio rapidly increases the risk of data collisions, since the sending neighbors are potentially hidden from each other. In this example we assume a fixed traffic flow, and with our global view we can easily adapt the wake up configuration to match the network's optimal data-per-wakeup ratio. But what if the amount of traffic is not fixed?

Event-driven networks can lie dormant for extended periods of time, and then suddenly generate a burst of traffic. If we configure the network to wake up often enough to cope with a traffic burst, we waste energy in between events. Conversely, if we configure the network for the low amount of traffic in between events, we risk network congestion and data losses when an event occurs. Several different solutions to the wake up configuration problem have been proposed, such as adapting wake up rates according to the detected traffic [3, 113]. A fundamental problem with adaptive protocols is that collisions must occur before the protocol can start adapting, and bursts are often short.

We address the problem of data collisions differently from adaptive protocols. Instead of adapting the wake up rate to reduce collisions, we make the network robust against collisions. Each device should be able to efficiently receive multiple data packets in a single wake up. Furthermore, the hidden terminal problem must be avoided as the sender nodes may be hidden to each other. Traditional Request-To-Send/Clear-To-Send [10] (RTS/CTS) protocols do not avoid the hidden terminal problem in these scenarios as the RTS packets collide.

We present Strawman in Paper E. Strawman is a contention resolution mechanism designed for receiver-initiated low-power protocols. Strawman is designed for networks that have a steady traffic flow but that may experience sudden traffic bursts. The mechanism efficiently handles the hidden terminal problem. We further implement and evaluate Strawman in a large-scale testbed in Paper F. We have also studied Strawman from a theoretical perspective in a paper that is not included in this thesis [47].

3. Contributions

This thesis makes three main scientific contributions. In addition, the thesis work has produced software that is available via the Contiki operating system and is used in both industry and the research community.

3.1 Scientific Contributions

The thesis makes three scientific contributions in the area of low-power wireless networking.

3.1.1 Timing-Accurate Simulation

I show that timing accuracy in simulation is important to achieve high performance when developing low-power wireless protocols. I present the Cooja simulation environment that allows for timing-accurate simulation. Using Cooja I have developed low-power wireless protocols that improve bulk transfer throughput [Paper B], convergecast latency [Paper C], and radio duty-cycling efficiency [Paper E, Paper F].

Timing-accurate simulation of both software execution and communication is important. Timing-accurate software execution revealed that packet copying was a major throughput bottleneck in bulk transfer protocols [Paper B]. This observation led to the development of the Conditional Immediate Transmission technique. Timing-accurate communication exposed that bursty traffic aggravates the hidden terminal problem in duty-cycled networks [Paper E]. This led us to develop Strawman that efficiently copes with hidden terminals.

3.1.2 Conditional Immediate Transmission

I present the Conditional Immediate Transmission (CIT) technique that can move packet copying off the critical path in low-power wireless protocols, thereby increasing throughput and lowering latency. With CIT, we reach 97% of the theoretical maximum throughput in a bulk transfer scenario in Paper B, and implement low-latency convergecast in Paper C.

We have further used insights from CIT to increase throughput over lossy links [32] and others have used CIT to build the pipelined TDMA-based bulk transfer protocol PIP [95]. For example, the Flush bulk transfer protocol de-

veloped without CIT achieves a throughput of 10kbit/s [64], whereas the more recent PIP protocol that uses CIT achieves 58kbit/s.

3.1.3 Duty Cycling with Bursty Traffic

I show that bursty traffic in duty-cycled low-power wireless networks aggravates the hidden terminal problem. I present the Strawman contention resolution mechanism [Paper E, Paper F] that is designed for networks that experience sudden traffic bursts. Strawman mitigates the hidden terminal problem by an efficient Request-To-Send/Clear-To-Send mechanism. We experimentally show that Strawman can increase the goodput with 77% in a traffic-saturated network when compared with a scalable random backoff-based mechanism that does not cope with hidden terminals.

Strawman makes networks robust against varying traffic levels and may in future work reduce the need for specialized traffic-adaptive protocols as well as for duty-cycling protocols that are tailored specifically for bursty traffic. Moreover, the Strawman technique may also be applicable in other types of networks, such as WiFi [107].

3.2 Software

I have developed the Cooja simulation environment and made it available for use via the Contiki operating system. Cooja has been the default simulation environment for Contiki since version 2.0 year 2006. Cooja has as a result been used by others to increase development efficiency [9, 24, 32], conduct node-level software experiments [97, 108, 110, 115], and conduct network-level experiments [60, 75, 89, 90, 117].

I am a member of the Contiki core team, where I developed and administer Contiki's nightly tests suite based on Cooja. I have ported Contiki functionality to new hardware platforms, and have fed back code into Contiki from many of my projects: checkpointing, Conditional Immediate Transmission, and Strawman.

4. Included Papers

4.1 Paper A

F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Tampa, Florida, USA, November 2006

Summary

This paper leverages my preceding master's thesis on a sensor network simulator for the Contiki operating system. There already existed a number of published sensor network simulators at the time, which were either not available to the public, or too tightly targeted a specific purpose such as a particular network protocol, or a particular radio environment. In contrast to previous work, we wanted a simulator primarily for understanding and developing sensor network applications, not only for experimenting with them. One of the main design goals was flexibility: the simulator was designed to allow adding new simulated radio mediums, plugins (simulator programs), and node peripherals.

We eventually developed what would become the main topic of Paper A: the cross-level simulation concept. Cross-level simulation allows different network nodes to be simulated with different levels of detail. Our main insight was that during development and debugging phases, developers require a very high level of detail, but typically only regarding a small subset of the simulated network. With cross-level simulation we could now simulate network routers and data sinks in high detail, whereas the often less error-prone data generators were simulated at a higher level of abstraction. The primary purpose of cross-level simulation was simulation performance: higher abstraction levels require less memory and less processing resources.

Reflections

The cross-level simulation concept is still central to Cooja, and has proven useful not only regarding simulation performance. Paper A presents three different detail levels: application, operating system, and emulation. The application level allows the developer to quickly prototype and test network protocols and applications, but has also extensively been used to simulate other types of entities that affect or observe the network execution but that are not necessarily themselves network members. Application nodes have for exam-

ple represented radio interference from nearby WiFi base stations [13], robots in remotely connected robot testbeds [23], and sheep that move according to real-world traces [89].

Cross-level simulation also fits nicely with the traditional develop-simulate-deploy development phases, which in Cooja becomes develop-simulate-emulate-deploy.

My Contributions

I implemented the Cooja simulator and its cross-level support. Joakim Eriksson provided an early version of MSPsim as backend for the emulated level. I carried out the experiments. I wrote the paper, with help and improvements from the co-authors.

4.2 Paper B

F. Österlind and A. Dunkels. Approaching the Maximum 802.15.4 Multi-hop Throughput. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, Charlottesville, Virginia, USA, June 2008

Summary

This paper was motivated by the batch-and-send paradigm [33] and advances in sensor network bulk transfer protocols [64]. In batch-and-send networks, individual sensors gather large amounts of sensor data before the data is sent to the network sink. This has two benefits: 1) the data-per-packet ratio is increased, and 2) the overall energy consumption is reduced by temporarily disabling the radio duty-cycling protocols, enabling high throughput bulk transfers. Since all network nodes that participate in a bulk transfer keep their radios turned on, the energy consumption is hence directly related to what throughput the bulk transfer protocol can achieve.

We observed a large discrepancy between the raw radio bitrate (250kbit/s) and state-of-the-art bulk transfer protocol throughput (10kbit/s) [64]. This paper sets out to determine the maximum achievable throughput in IEEE 802.15.4, and what factors affect and limit the throughput.

We found that packet copying was a significant throughput bottleneck, and introduced a novel technique called *Conditional Immediate Transmission*. With Conditional Immediate Transmission, we remove packet copying from the critical path, which—on our hardware platform—nearly doubled the bulk transfer throughput.

Reflections

The impact of packet copying largely depends on the hardware platform configuration. Our experiments were performed on the Tmote Sky device, on

which communication between the microcontroller and the radio chip is via an SPI bus. The SPI bitrate depends on the microcontroller's clock frequency, which can be increased, thus lessening the impact of packet copying on bulk transfer throughput.

The main advantage of Conditional Immediate Transmission, however, does not depend on the packet copying bitrate. The main advantage of Conditional Immediate Transmission is its deterministic behavior: protocols can now rely on deterministic low-latency responses, irrespectively of remote packet copying bitrates or packet sizes. We use the deterministic properties of Conditional Immediate Transmission in both Paper C and Paper E.

Our technique for removing packet copying off the critical path was adopted by others to implement a full-scale bulk transfer protocol [95], to decrease the slot lengths in a TDMA protocol [45], and was later also used by us to implement high-throughput transfer in lossy environments [32].

My Contributions

I invented Conditional Immediate Transmission, and implemented it for low-power wireless networks. I structured and wrote the initial draft of the paper. The final text had significant contributions from Adam Dunkels. I performed the experiments.

4.3 Paper C

H. Zhang, F. Österlind, P. Soldati, T. Voigt, and M. Johansson. Rapid Convergecast on Commodity Hardware: Performance Limits and Optimal Policies. In *The IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON)*, Boston, Massachusetts, USA, June 2010

Summary

This paper leverages insights from Paper B that moved packet-copying off the critical path in a bulk transfer scenario. In contrast to Paper B, we now consider a convergecast scenario. Our intended application is industrial with control loops over low-power wireless networks. The network response latency is hence of major importance.

The main insight in this work is that packet-copying should be performed in a separate time slot, resulting in three different time slot types: transmission, reception, and packet-copying.

Our proposed convergecast protocol is synchronized and uses both frequency-hopping and separated packet-copying time slots. We present both a centralized and a distributed algorithm, and prove that the algorithms produce time-optimal schedules.

Reflections

We used Cooja both during the development of this protocol, and for setting up the experiments. For experiments, we first configured a simulated network that matched our hardware testbed, verified the protocol behavior in simulation, and then deployed the code to the testbed and measured the performance.

The development and debugging of low-level, high-performance protocols like in this work can be very time-consuming and difficult. To assist me in understanding the behavior, I developed a Timeline extension to Cooja. The Timeline was later made available to the public and is now one of the most appreciated features in Cooja [86].

My Contributions

I developed the separated copying slots technique together with the other authors. I implemented the protocol in Contiki, and carried out the performance experiments in both simulation and on real hardware. The algorithm for deriving an optimal schedule was developed by Haibo Zhang and the other co-authors.

4.4 Paper D

F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet Checkpointing: Enabling Repeatability in Testbeds and Realism in Simulations. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Cork, Ireland, February 2009

Summary

This work was motivated by an ongoing debate on how simulators should be used to perform research experiments, or whether the simplistic radio models found in low-power wireless simulation disqualify them altogether. My opinion is that the experiment alone determines whether a simulation-based experiment is appropriate or not; in some experiments extremely simple ideal models are preferred over complex and realistic models—an opinion which is reflected by Cooja's flexible design.

A number of papers proposed mixing simulation with testbeds to introduce realism in (partially) simulated experiments [76, 88, 124, 125]. The simulation environment is connected to a hardware testbed, and the network is partly simulated and partly run on real hardware. This technique is called hybrid simulation. A problem with hybrid simulation, as further discussed in this paper, is that many of the beneficial properties of simulation alone disappear including repeatability and non-intrusive visibility.

In this paper, in contrast to hybrid simulation, we instead propose mixing testbeds and simulation over the *temporal* axis. The entire network is hence both represented in the simulation and in the testbed. The network state and

execution, however, moves between the two domains. This approach provides repeatability and non-intrusive visibility while the network is in simulation, and realism while in testbed. This paper proposes and evaluates the basic mechanism.

Reflections

Sensornet checkpointing is an extremely powerful technique, but requires sophisticated infrastructure support. The dependencies include a testbed of Tmote Sky devices, testbed checkpointing software support, the Cooja simulator that can emulate the Tmote Sky device, a Contiki checkpointing library, and a middleware for transferring network checkpoints between testbed and simulation. These prerequisites make sensornet checkpointing difficult to adopt for new users, as sensornet checkpointing has a relatively large setup time compared to for instance timing-accurate simulation.

My Contributions

I implemented the checkpointing server-side software, and the Contiki library and its port to the Tmote Sky device. I wrote the paper, and carried out the experiments. Joakim Eriksson provided bug fixes and feature requests for the MSPsim emulator. Nicolas Tsiftes provided the run-length encoding experiment results.

4.5 Paper E

F. Österlind, N. Wirström, N. Tsiftes, N. Finne, T. Voigt, and A. Dunkels. StrawMAN: Making Sudden Traffic Surges Graceful in Low-Power Wireless Networks. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, Killarney, Ireland, June 2010

Summary

This paper presents Strawman, a contention resolution mechanism for low-power wireless networks. Strawman makes networks robust to sudden increases of traffic, thus allowing a network to be configured for the steady-state network traffic instead of the anticipated traffic peaks.

One of the major energy consumers in radio duty-cycled networks is idle listening: waking the radio up to check for new incoming data when there is no new data generated in the network. If the network is configured to use very low duty cycles, however, it becomes sensitive to traffic bursts and hidden terminals. Moreover, the traditional technique to handle the hidden terminal problem, Request-To-Send/Clear-To-Send (RTS/CTS), has been shown to induce a prohibitively high overhead in these networks [93].

Strawman builds upon Low-Power Probing (LPP) [81, 112], and was inspired by the Pollcast [25] and the Backcast [34] communication primitives.

LPP makes receivers probe for incoming data by a radio transmission; communication is receiver-initiated. Pollcast and Backcast both make explicit use of radio packet collisions.

Strawman is conceptually similar to both random backoff and to bit-dominance protocols [100]. In contrast to random backoff where contenders back off for a random duration, Strawman contenders transmit a random-length request packet which length determines whether they win channel access or not. In contrast to bit-dominance protocols, Strawman uses dynamic priorities and does not require on-off-keying radio modulation.

Reflections

Strawman is a technique that can significantly affect how low-power wireless networks applications are designed, as it makes the network robust against traffic bursts. This paper proposes the basic technique, but only provides initial experimental results. Before the technique can be adopted by the general public, we need to evaluate it in real deployments. By contrast, the de facto standard technique random backoff has been well-tested in the Internet for decades. Another important factor is code availability; we plan to release Contiki's Strawman implementation with the next release of Contiki. Paper F further develops Strawman.

My Contributions

I invented the Strawman mechanism, and implemented it in Contiki. I wrote the paper with help from Adam Dunkels and Niklas Wirström. The name Strawman was suggested by Adam Dunkels. I carried out the experiments. Niclas Finne helped carry out the carrier detect sensitivity experiments in the background section. Niklas Wirström helped implement the software needed for the request resolution experiment.

4.6 Paper F

F. Österlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels. Strawman: Resolving Collisions Through Collisions. In submission

Summary

This paper further develops the Strawman contention resolution mechanism proposed in Paper E. We reimplement the Strawman mechanism for Contiki, and perform thorough experiments in a large-scale testbed, including packet-length estimation robustness, and how Strawman copes with saturated traffic and hidden terminals.

We show that packet-length estimation is robust to both the number of contenders and to external radio interference. We integrate Strawman with the Low-Power Probing protocol RI-MAC, and compare its performance with

backoff-based approaches. Our experiments further show that backoff-based approaches relies heavily on capture effect, which allows a radio to correctly receive a radio packet even when other colliding radio packets are transmitted simultaneously [72].

Reflections

In contrast to the previous Strawman workshop paper, this paper develops and evaluates a complete Strawman implementation in a large-scale testbed. During this process we saw the need for, and developed, many features that are now central to the Strawman protocol. For instance, both the use of a non-uniform distribution and Strawman's multi-channel operation were motivated by early experimental results from the TWIST testbed. For me, these discoveries stress the importance of actually implementing and evaluating applications and protocols on real hardware and in large-scale testbeds.

My Contributions

I wrote the paper with help from the co-authors, and especially from Luca Mottola who helped structure and write about the experiments and the related work. I implemented Strawman and conducted all experiments.

5. Related Work

This thesis argues for timing-accurate simulation when developing low-power wireless protocols. Related work includes simulation tools as well as alternative development methods that may be combined with, or altogether replace, simulation. Related work further includes low-power wireless protocols and mechanisms for achieving high throughput and low power consumption.

5.1 Low-Power Wireless Simulation

A simulation contains a model of a system, where the model is typically too complex to theoretically analyze. The amount of details in the simulated model varies between different simulation environments and affects both the simulation performance and applicability. A simulation with few details requires less memory and executes faster than a simulation with a high level of details, whereas the detailed simulation may better represent a real-world system. In the context of low-power wireless networks, a simulation contains models of the radio environment, the sensor data environment, and the device applications. The devices are typically modeled at either of three detail levels: the network, the operating system, or the hardware. By contrast, Cooja supports simulation at all three levels, as described next in Section 5.1.4. A few well-known simulators that simulate networks at a single detail level are reviewed below.

5.1.1 Network-Level

Network-level simulations enable study of how network protocols behave given different stimuli and over different network topologies. The network-level is well suited for early prototyping.

The perhaps most well-known network-level simulator is ns-2 [121]. ns-2 contains a large set of pre-installed network and Medium Access Control (MAC) protocols and radio environment models to choose from, and allows a user to quickly configure a new network simulation.

Several extensions have been added to ns-2 to add or improve support for simulation of low-power wireless networks [88, 128]. Wittenburg and Schiller added support for simulating real deployable sensor network application code in ns-2 [128]. SensorSim extends ns-2 with battery and sensor models [88].

Other sensor network simulators at the network-level include SENSE [19], SENS [114], Castalia [14] and MiXiM [67] that build upon OMNeT++ [22], and QualNet [1] that builds upon GloMoSim [133].

Network-level simulators typically do not simulate deployable application code. Even simulators that support deployable code, such as the ScatterWeb simulator [128], only model the application and not the complete operating system. This limits the applicability of network-level simulators.

5.1.2 Operating System-Level

Simulation at the operating system-level is more detailed than the network-level. An OS-level simulation exercises the complete operating system, including the network protocols, memory management libraries, and the process scheduler.

OS-level simulation typically employs glue drivers to replace the hardware peripheral drivers, such as the radio driver. The operating system together with the simulated application is then compiled and executed during the simulation. OS-level simulation offers a good tradeoff between simulation execution speed and accuracy.

Two examples of OS-level simulation environments are TOSSIM [73] that simulates TinyOS, and EmSim [48] that simulates the EmStar environment.

5.1.3 Hardware-Level

The hardware detail level models the entire low-power wireless device, including real code application, operating system, and all hardware peripherals on the device. The simulation environment can load pre-compiled binaries for the modeled hardware platform, and thus supports simulating the exact same binary as is uploaded to real hardware. Simulation at the hardware level offers high details and accurate execution timing at the price of higher execution and memory overhead.

Sensor device hardware emulators include Avrora [119] that emulates AVR-based sensor node hardware, ATEMU [63] that emulates the MICA2 sensor platform, and MSPsim [38] that emulates the TI MSP430 microcontroller and the Tmote Sky device [79]. AvroraZ [4] is an extension to Avrora that adds emulation of the CC2420 radio and the MicaZ platform.

The Worldsens development environment [44] is structurally similar to Cooja/MSPsim. The network simulator WSNet in Worldsens interconnects several WSim instances, similarly to how Cooja connects several MSPsim instances. WSim, like MSPsim, emulates instances of MSP430-based low-power devices.

5.1.4 Mixing Levels

The amount of details modeled in a simulation environment greatly affects both the simulator applicability and performance. The three detail levels discussed above all have individual strengths and weaknesses. For example, simulations with fewer details have less restrictions and run faster compared to more detailed simulations. In addition, a high-level simulation is not restricted by hardware limitations or low-level implementation details. By contrast, detailed hardware-level simulations have the exact same memory and processing restrictions as the real hardware devices, which simplifies migrating from simulation to real hardware.

The concept of cross-level simulation is introduced in Paper A. Cross-level simulation allows for simulating different devices with different amounts of detail. The main insight behind cross-level simulation is that a developer is often interested in only a subset of nodes in a simulated network, such as the nodes that are closest to the data sink in a collection network. For example, a cross-level simulation may contain a few tens of emulated devices that are positioned close to the data sink, and a few hundred OS-level devices further from the sink.

Hybrid simulation, also called co-simulation, connects real hardware devices with simulated ones [76, 88, 111, 124, 125]. By mixing simulated and real hardware, hybrid simulation is conceptually similar to Hardware-in-the-loop (HIL) simulation. Simulators that make use of hybrid simulation environments strive to either provide higher scalability by the introduction of simulated nodes [125], higher realism by the introduction real hardware nodes [76, 88], or for improved debugging and testing support [124]. The connection between simulated and real nodes is typically a serial port. Another type of hybrid simulation gathers information from real networks to enhance realism in simulated networks. For instance, the Cooja extension RealSim by Strube et al. updates the simulated radio environment in real-time according to measurements in a real-hardware testbed [111].

We propose sensornet checkpointing in Paper D, a technique that is similar to hybrid simulation as it involves both real hardware and simulated nodes. Sensornet checkpointing allows transferring a complete network state between simulation and real hardware—the network execution is either in simulation or on hardware, not in both as with hybrid simulation.

5.1.5 Modeling Time

How simulation environments model time affects both performance and accuracy of experiments, and several different approaches have been proposed [18, 44, 70, 105, 119]. TrueTime is a simulator for networked real-time control systems, and simulates how device constraints like microcontroller CPU speed affect an application [18]. TrueTime is an application-level simulator, where

application tasks are implemented in MATLAB. TrueTime cannot simulate real deployable microcontroller code, and targets early development phases.

PowerTOSSIM [105] and TimeTOSSIM [70] are extensions to the TinyOS simulator TOSSIM. Like TOSSIM they are OS-level simulators but they add additional timing accuracy by code analysis and instrumentation. Both PowerTOSSIM and TimeTOSSIM instrument the simulated TinyOS system with estimates of code execution durations. They are less detailed than device emulators and offer a trade-off between timing accuracy and simulation performance. The lack of details limits their applicability for development of low-power wireless protocols. For example, PowerTOSSIM and TimeTOSSIM do not model peripherals with high timing such as the communication bus between the microcontroller and the radio transceiver. But such detailed models were needed to gain the insights behind the Conditional Immediate Transmission technique in Paper B.

Avrora [119] and Worldsense [44] are hardware-level simulators and, like MSPsim, offer cycle-accurate timing details. In this thesis I demonstrate that this high level of timing-accuracy is important to develop low-power wireless protocols with high performance.

5.2 Development Methods

Development of low-power wireless networks is difficult due to the distributed applications and severe resource-limitations. Simulation is often used as it offers full visibility, repeatability, and control of the simulated network. To simplify development on real hardware, techniques and tools that provide similar functionality as available in simulation have been proposed.

5.2.1 Repeatability in Testbeds

Hardware testbeds allow for uploading and running low-power wireless applications on real hardware in controlled environments. Well-known research testbeds include the publicly available 200-node TWIST testbed [51] and Harvard's 190-nodes MoteLab testbed [126]. Simulated network mobility is achieved with tools such as BonnMotion [7, 8], whereas real-hardware mobility can be implemented using robot-equipped testbeds [62, 98].

Repeatability of hardware experiments was addressed by Luo et al. that present EnviroLog [78]. EnviroLog records and later replays all asynchronous events on a sensor device, to enable performance evaluations on real-hardware networks.

5.2.2 Non-Intrusive Visibility

A number of approaches have been proposed to increase the visibility into low-power networks, both simulated and hardware-based. Wachs et al. argue for building in visibility directly into the network protocols, and introduce a new energy cost-based visibility metric [123].

Römer presents passive distributed assertions (PDAs) [102]. PDAs allows for non-intrusively exposing internal application state in the network by instrumenting the application pre-deployment to periodically transmit application state. A temporary sniffer network is then deployed next to the monitored network. PDAs are non-intrusive in the sense that the application is not affected by the addition or removal of the sniffer network. Sympathy by Ramanathan et al. combines both passively collected information with active periodic beacons to detect network faults [96].

Software-based [31, 42] and hardware-based [61] power consumption monitors measure the power performance of deployed applications, and thus provide visibility into a deployed application's power profile.

The Aveksha system by Tancreti et al. uses the debug module of the on-board microcontroller to non-intrusively observe the execution of a deployed application [115]. Using Aveksha, a watchpoint-based user interface similar to the simulated Cooja Timeline [86] can be provided, but Aveksha provides visibility into real-hardware devices which Cooja cannot.

5.2.3 Source-level Debugging

Source-level debuggers are popular tools to debug and inspect both simulated and real-hardware networks. Huber et al. integrate YETI [16] with Cooja to simplify development of TinyOS-based applications [56]. YETI allows for debugging and monitoring of a simulated Cooja network via GDB interfaces. Similarly, the Clairvoyant source-level debugger allows for remote access into deployed hardware networks [130].

Marionette provides basic remote procedure calls (RPC) that enables a developer to login to devices in a deployed network [127]. The developer can interactively and remotely execute functions, and read and write variable states on the deployed devices.

5.3 Low-Power Wireless Protocols

We propose the Conditional Immediate Transmission (CIT) pipelining technique in Paper B, and the Strawman contention resolution mechanism for traffic-bursty radio duty-cycled networks in Paper E.

5.3.1 High-Throughput Protocols

The throughput in a low-power wireless network may directly affect the overall network power consumption. Power-saving mechanisms such as radio duty-cycling are often disabled during bulk transfers, so increased throughput allows the network to reactivate duty-cycling earlier [41].

We propose the Conditional Immediate Transmission (CIT) technique in Paper B. CIT removes packet copying off the critical path and is conceptually similar to no-copy APIs. No-copy APIs have been shown to increase performance in general purpose systems [2, 26]. CIT is a pipelining technique for building upper-layer protocols with high throughput, for example bulk transfer protocols. We demonstrate that CIT can significantly increase multi-hop bulk transfer throughput in Paper B and decrease convergecast latency in Paper C.

Bulk transfer protocols such as Flush [64], PIP [95], and Burst Forwarding [32] provide complete high-throughput data transport protocols. Flush is a bulk transfer protocol that provides end-to-end reliability and hop-by-hop rate control [64]. By contrast, the bulk transfer protocol we built in Paper B is best-effort, so their achieved throughputs should hence not be compared directly. Rather, CIT may be used by upper-layer protocols such as Flush to increase throughput.

PIP by Raman et al. use CIT to increase bulk transfer throughput in a reliable multi-channel TDMA protocol [95]. In addition, we have used insights from CIT in the Burst Forwarding protocol to provide high throughput over lossy links [32]. We have also built a convergecast protocol that separates packet copying from packet transmissions, to increase convergecast throughput and decrease latency in Paper C.

5.3.2 Coping with Traffic Bursts

Event-driven networks monitor the physical environment and only occasionally and suddenly generate large amounts of data in the network [59]. Strawman is a contention resolution mechanism that sits inside a receiver-initiated protocol, and is designed for networks that have a steady flow of data but that may experience sudden traffic bursts. For example, such a network may collect temperature measurements regularly, but would generate a large burst of alarm traffic from several network nodes upon detecting fire.

Coping with traffic bursts can be implemented at different levels of a system. ESRT [103] interacts between the application layer and the MAC layer to adapt to varying traffic levels. Similarly, Chi by Finne et al. reconfigures the entire network stack according to changes in network traffic by leveraging a separate system-wide configuration [41].

The Reliable Bursty Convergecast (RBC) protocol operates at the network and MAC layer and reliably collects bursts of data from an entire network to a single collection point [134].

Several MAC protocols that are specifically designed to cope with bursty traffic have been proposed [5, 57, 99, 101]. Z-MAC alternates between synchronized and contention-based operation depending on the amount of network traffic [99]. BurstMAC calculates and distributes ad-hoc per-round neighbor schedules to efficiently and quickly collect data transmission per neighbor [101].

Strawman is a contention resolution mechanism that sits inside a receiver-initiated radio duty-cycling protocol, and may be combined with many of the proposed upper-layer techniques for handling traffic bursts. Moreover, Strawman is activated only when needed, upon detecting data collisions, and has otherwise zero overhead.

5.3.3 Contention Resolution

Research on radio duty-cycling protocols has largely focused on the duty-cycling mechanism itself, and has assumed a random backoff-based contention resolution mechanism, such as exponential backoff [93, 129]. In networks with stable traffic patterns, contention and data collisions can be avoided by configuring or dynamically adapting the sleep cycles. In networks with bursty traffic, however, the overhead of configuring for worst-case traffic rates becomes prohibitively high. Jamieson et al. propose sampling random backoffs from a geometric distribution, for improved scalability and a bounded maximum backoff delay [59]. Strawman similarly relies on a non-uniform distribution [47].

5.3.4 Bit-Dominance Protocols and Radio Collisions

At the core of Strawman lies the ability to schedule channel access by measuring the longest of several colliding transmissions. Several other types of protocols rely on explicit radio collisions, such as bit-dominance protocols and protocols that exploit constructive interference.

Bit-dominance protocols schedule channel access by identifying a single highest prioritized sender among multiple colliding transmissions, and have been proposed for both the wired [118] and wireless domain [91, 100]. To be able to extract the dominant sender, bit-dominance protocols such as WiDom [91] and BitMAC [100] rely on On-Off-Keying (OOK).

The Black Burst protocol, like Strawman, does not rely on OOK modulation but instead measures the contention packets' lengths to determine a single channel winner [107]. In contrast to Strawman, Black Burst targets sender-initiated CSMA protocols, does not cope with hidden terminals, and is not designed for duty-cycled low-power networks.

Another set of protocols also rely on synchronized radio collisions, but do not extract information from the actual collisions, but rather from the existence of the collisions [17, 25, 34]. The MAC protocol FlipMAC [17] and the net-

work primitive Countcast [25] are both receiver-initiated and work by letting neighbors collide for several consecutive rounds, while counting the number collision rounds. Each neighbor randomly and locally decides whether to participate in the following round. Although the two protocols work similarly, they are used for different purposes. Countcast is used to estimate the number of neighbors, whereas FlipMAC is used to elect a single channel winner. The Backcast primitive [34] synchronizes several identical radio transmissions to achieve non-destructive interference at the receiver. Backcast is used in the radio duty-cycling protocol A-MAC to distinguish radio transmissions from background noise [35].

5.3.5 The Hidden Terminal Problem

The hidden terminal problem occurs when two or more network nodes transmit to a common destination, but fail to detect each other's transmissions and therefore cause data collisions [10]. Request-To-Send/Clear-To-Send (RTS/CTS) protocols are commonly used to mitigate the hidden terminal problem, and have been evaluated also in low-power wireless networks [131]. Traditional RTS/CTS protocols, however, have been shown to exhibit a high overhead in low-power networks [93].

Strawman shares many properties with RTS/CTS protocols, as is solves the hidden terminal problem via its receiver-originated decision transmission. In contrast to RTS/CTS protocols, Strawman allows for several simultaneous channel contenders, by first synchronizing all contenders with a probe transmission and then electing the winner by measuring who sent the longest RTS packet.

6. Conclusions

In this thesis I show that timing-accurate simulation is important to improve performance in low-power wireless protocols. High timing accuracy becomes important due to the extremely timing-sensitive radio duty-cycling techniques that are employed to preserve power coupled with the low resources available on low-power devices.

I develop the simulation environment Cooja that when combined with the MSPsim device emulator simulates low-power wireless networks with high timing accuracy. Using Cooja, I demonstrate how timing-accurate simulation is used to facilitate development of low-power wireless protocols. I present the Conditional Immediate Transmission technique that removes packet copying off the critical path and the contention resolution mechanism Strawman that handles sudden traffic bursts and copes with hidden terminals.

I present the cross-level simulation and the sensornet checkpointing techniques. These techniques improve scalability and realism of low-power wireless network experiments, but my experience is that the availability of timing-accurate simulation reduces the need for techniques such as cross-level simulation and sensornet checkpointing when developing low-power wireless protocols.

The development of Cooja is still active. Current projects add support for new types of emulated low-power devices, make the simulation engine more efficient, and enable high-coverage testing using sensornet checkpointing. The Cooja simulation environment is used by both academia and industry and is the default simulator in the Contiki operating system since year 2006.

Bibliography

[1] Q.N. Simulator. Scalable Network Technologies. Web page, 2011. URL `http://www.scalable-networks.com/`. Visited 2011-09-28.

[2] B. Ahlgren, P. Gunningberg, and K. Moldeklev. Increasing Communication Performance with a Minimal-Copy Data Path Supporting ILP and ALF. In *Journal of High Speed Networks*, 1996.

[3] G.-S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo. Funneling-mac: A localized, sink oriented mac for boosting fidelity in sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Boulder, Colorado, USA, November 2006.

[4] R. P. Alberola and D. Pesch. AvroraZ: Extending Avrora with an IEEE 802.15.4 Compliant Radio Chip Mode. In *3rd ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks*, Vancouver, Canada, October 2008.

[5] M. Anwander, G. Wagenknecht, T. Braun, and K. Dolfus. Beam: A burst-aware energy-efficient adaptive mac protocol for wireless sensor networks. In *International Conference on Networked Sensing Systems (INSS)*, 2010.

[6] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5), December 2004.

[7] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010.

[8] N. Aschenbruck, R. Ernst, M. Schwamborn, F. Österlind, and T. Voigt. Adding mobility to wireless sensor network simulations. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, 2010.

[9] S. Bhadra, S-H. Choi, Y. Sun, and X. Lu. Demo abstract: Achieving a 10x lifetime increase with ieee 802.15.4e motes. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, November 2011.

[10] V. Bharghavan, A. Demers, S. Schenker, and L. Zhang. Macaw: a media access protocol for wireless lans. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM)*, London, UK, 1994.

[11] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. *CM/Kluwer Mobile Networks & Applications (MONET), Special Issue on Wireless Sensor Networks*, 10(4), August 2005.

[12] C. A. Boano, Z. He, Y. Li, T. Voigt, M. Zuniga, and A. Willig. Controllable Radio Interference for Experimental and Testing Purposes in Wireless Sensor Networks. In *Proceedings of the 4th International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Zurich, Switzerland, October 2009.

[13] C. A. Boano, K. Römer, F. Österlind, and T. Voigt. Demo Abstract: Realistic Simulation of Radio Interference in COOJA. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Bonn, Germany, 2011.

[14] A. Boulis. Castalia: Revealing Pitfalls in Designing Distributed Algorithms in WSN. In *Poster proceedings of the 5th international conference on Embedded networked sensor systems*, pages 407–408, 2007.

[15] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Boulder, Colorado, USA, 2006.

[16] N. Burri, R. Schuler, and R. Wattenhofer. Yeti: A tinyos plug-in for eclipse. In *Organization of the Workshop on Real-World Wireless Sensor Networks (REALWSN)*, 2009.

[17] D. Carlson and A. Terzis. Flip-mac: A density-adaptive contention-reduction protocol for efficient any-to-one communication. In *Proceedings of Distributed Computing in Sensor Systems (DCOSS)*, Barcelona, Spain, 2011.

[18] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K-E Årzén. How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime. In *Proceedings of IEEE Control Systems Magazine*, June 2003.

[19] G. Chen, J. Branch, M.J. Pflug, L. Zhu, and B. Szymanski. Sense: A sensor network simulator. In *Advances in Pervasive Computing and Networking*, 2004.

[20] Chipcon AS. CC2420 Datasheet (rev. 1.3), 2005. URL `http://www.chipcon.com/`.

[21] Cisco. Cisco, Atmel and the Swedish Institute of Computer Science (SICS) Collaborate to Support a Future Where Any Device Can Be Connected to the Internet. Press release, October 2008.

[22] U.M. Colesanti, C. Crociani, and A. Vitaletti. On the accuracy of omnet++ in the wireless sensor networks domain: simulation vs. testbed. In *Proceedings of the 4th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 25–31, Chania, Greece, October 2007.

[23] CONET. Cooperating Objects Network of Excellence (CONET). Web page. URL `http://www.cooperating-objects.eu/`. Visited 2011-06-16.

[24] M. Daum. *Verteilung globaler Anfragen auf heterogene Stromverarbeitungssysteme*. PhD dissertation, Der Technischen Fakultät der Universität Erlangen-Nürnberg, 2011.

[25] M. Demirbas, O. Soysal, and M. Hussain. Singlehop collaborative feedback primitive for wireless sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2008.

[26] P. Druschel and L. Peterson. Fbufs: a high-bandwidth cross-domain transfer facility. In *Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 189–202, Asheville, North Carolina, United States, 1993.

[27] A. Dunkels. The Contiki Operating System. Web page. URL `http://www.sics.se/contiki/`. Visited 2009-08-03.

[28] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, USA, May 2003.

[29] A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne. The announcement layer: Beacon coordination for the sensornet stack. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Bonn, Germany, 2011.

[30] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, November 2007.

[31] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, Cork, Ireland, June 2007.

[32] S. Duquennoy, F. Österlind, and A. Dunkels. Lossy Links, Low Power, High Throughput. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, November 2011.

[33] P. Dutta, D. Culler, and S. Shenker. Procrastination might lead to a longer and more useful life. In *Proceedings of the Workshop on Hot Topics in Networks (ACM HotNets)*, Atlanta, GA, USA, November 2007.

[34] P. Dutta, R. Musaloiu-E., I. Stoica, and A. Terzis. Wireless ack collisions not considered harmful. In *Proceedings of the Workshop on Hot Topics in Networks (ACM HotNets)*, 2008.

[35] P. Dutta, S. Dawson-Haggerty, Y. Chen, C-J M. Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, November 2010.

[36] A. El-Hoiydi, J.-D. Decotignie, C. C. Enz, and E. Le Roux. Poster Abstract: WiseMAC, an Ultra Low Power MAC Protocol for the WiseNET Wireless Sensor Network. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2003.

[37] Ericsson. CEO to shareholders: 50 billion connections 2020. Press release, April 2010. URL `http://www.ericsson.com/thecompany/press/releases/2010/04/1403231`.

[38] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. Mspsim – an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Delft, The Netherlands, January 2007.

[39] European Commission, Cluster of European Research Projects on the Internet of Things. Vision and Challenges for Realising the Internet of Things. Book, March 2010.

[40] L. M. Feeney and D. Willkomm. Energy framework: An extensible framework for simulating battery consumption in wireless networks. In *3rd International Workshop on OMNeT++ in conjunction with 3rd International Conference on Simulation Tools and Techniques Simutools*, March 2010.

[41] N. Finne, J. Eriksson, N. Tsiftes, A. Dunkels, and T. Voigt. Improving sensornet performance by separating system configuration from system logic. In *Proceedings of the Sixth European Conference on Wireless Sensor Networks (EWSN2010)*, Coimbra, Portugal, February 2010.

[42] R. Fonseca, P. Dutta, P. Levis, and I. Stoica. Quanto: Tracking energy in networked embedded systems. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, pages 323–338, 2008.

[43] HART Communication Foundation. Wireless HART Technology. Web page, June 2011. URL `http://www.hartcomm.org/protocol/wihart/wireless_technology.html`. Visited 2011-06-16.

[44] A. Fraboulet, G. Chelius, and E. Fleury. Worldsens: Development and Prototyping Tools for Application Specific Wireless Sensors Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, Cambridge, Massachusetts, USA, 2007.

[45] V. Gabale, B. Raman, K. Chebrolu, and P. Kulkarni. Lit mac: Addressing the challenges of effective voice communication in a low cost, low power wireless mesh network. In *Proceedings of the First ACM Symposium on Computing for Development*, page 5. ACM, 2010.

[46] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. 2002.

[47] E. Ghadimi, P. Soldati, F. Österlind, H. Zhang, and M. Johansson. Hidden terminal-aware contention resolution with an optimal distribution. In *The Eighth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2011.

[48] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. EmStar: A Software Environment for Developing and Deploying Wireless Sensor Networks. In *Proceedings of the USENIX Annual Technical Conference*, 2004.

[49] L. Gu and J. Stankovic. Radio-triggered wake-up capability for sensor networks. In *Proceedings of RTAS*, 2004.

[50] C. Han, R. K. Rengaswamy, R. Shea, E. Kohler, and M. Srivastava. SOS: A dynamic operating system for sensor networks. In *Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Seattle, WA, USA, 2005.

[51] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality (REALMAN'06)*, 2006.

[52] Harvard Sensor Networks Lab. CodeBlue: Wireless Sensors for Medical Care. Web page. URL `http://fiji.eecs.harvard.edu/CodeBlue`. Visited 2011-09-28.

[53] M. Hefeeda and M. Bagheri. Forest fire modeling and early detection using wireless sensor networks. In *Ad Hoc & Sensor Wireless Networks*, 2008.

[54] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, 2001.

[55] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, November 2000.

[56] R. Huber, P. Sommer, and R. Wattenhofer. Demo abstract: Debugging wireless sensor network simulations with yeti and cooja. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, 2011.

[57] P. Hurni and T. Braun. Maxmac: A maximally traffic-adaptive mac protocol for wireless sensor networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Coimbra, Portugal, February 2010.

[58] IEEE. 802.11g. IEEE Standard for Information technology, October 2010.

[59] K. Jamieson, H. Balakrishnan, and Y. C. Tay. Sift: a MAC Protocol for Event-Driven Wireless Sensor Networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Zurich, Switzerland, February 2006.

[60] A. Jhumka and L. Mottola. On Consistent Neighborhood Views in Wireless Sensor Networks. In *28th IEEE International Symposium on Reliable Distributed Systems (SRDS)*, 2009.

[61] X. Jiang, P. Dutta, D. Culler, and I. Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *Proceedings*

of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN), Cambridge, Massachusetts, USA, 2007.

[62] D. Johnson, T. Stack, R. Fish, D. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[63] M. Karir. atemu - Sensor Network Emulator / Simulator / Debugger. Technical report, Center for Satellite and Communication Networks, Univ. of Maryland, 2003.

[64] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: A reliable bulk transport protocol for multihop wireless networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, November 2007.

[65] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, M. Durvy, JP Vasseur, A. Terzis, A. Dunkels, and D. Culler. Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, 2011.

[66] I. Konovalov, J. Neander, M. Gidlund, F. Österlind, and T. Voigt. Evaluation of WirelessHART Enabled Devices in a Controlled Simulation Environment. In *IEEE International Symposium on Industrial Electronics*, Gdansk, Poland, 2011.

[67] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P.T. K. Haneveld, T.E.V. Parker, O.W. Visser, H.S. Lichte, and S. Valentin. Simulating Wireless and Mobile Networks in OMNeT++. The MiXiM vision. In *Proceeding of the 1st International Workshop on OMNeT++*, March 2008.

[68] G. Kunz, O. Landsiedel, and G. Wittenburg. *Book chapter: From Simulations to Deployments. In: Modeling and Tools for Network Simulation*. Springer, April 2010.

[69] S. Kurkowski, T. Camp, and M. Colagrosso. MANET Simulation Studies: The Incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, October 2005.

[70] O. Landsiedel, H. Alizai, and K. Wehrle. When Timing Matters: Enabling Time Accurate and Scalable Simulation of Sensor Network Applications. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, St. Louis, MO, USA, 2008.

[71] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, April 2006. IEEE.

[72] K. Leentvaar and J. Flint. The capture effect in FM receivers. *Communications, IEEE Transactions on*, 24(5):531–539, 1976. ISSN 0090-6778.

[73] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Los Angeles, California, USA, 2003.

[74] C. Liang, N. Priyantha, J. Liu, and A. Terzis. Surviving wi-fi interference in low power zigbee networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, November 2010.

[75] A. Lindgren, C. Mascolo, M. Lonergan, and B. McConnell. Seal-2-seal: A delay-tolerant protocol for contact logging in wildlife monitoring sensor networks. In *Mobile Ad Hoc and Sensor Systems (MASS)*, 2008.

[76] S. Lo, J. Ding, S. Hung, J. Tang, W. Tsai, and Y. Chung. SEMU: A Framework of Simulation Environment for Wireless Sensor Networks with Co-simulation Model. In *Proceedings of International Conference on Grid and Pervasive Computing (GPC), Lecture Notes in Computer Science (LNCS*, 2007.

[77] M. Lunden and A. Dunkels. The Politecast Communcation Primtive for Low-Power Wireless. *The ACM SIGCOMM Computer Communications Review*, April 2011.

[78] L. Luo, T. He, G. Zhou, L. Gu, T. F. Abdelzaher, and J. A. Stankovic. Achieving repeatability of asynchronous events in wireless sensor networks with envirolog. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[79] MoteIV. Tmote Sky Low Power Wireless Sensor Module. Datasheet, June 2006.

[80] Municipal Transportation Agency. SFpark. Web page. URL `http://sfpark.org`. Visited 2011-09-28.

[81] R. Musaloiu-E., C-J. M. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, St. Louis, Missouri, USA, 2008.

[82] F. Österlind and A. Dunkels. Approaching the Maximum 802.15.4 Multi-hop Throughput. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, Charlottesville, Virginia, USA, June 2008.

[83] F. Österlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels. Strawman: Resolving Collisions Through Collisions. In submission.

[84] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Tampa, Florida, USA, November 2006.

[85] F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet Checkpointing: Enabling Repeatability in Testbeds and Realism in Simulations. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Cork, Ireland, February 2009.

[86] F. Österlind, J. Eriksson, and A. Dunkels. Demo Abstract: Cooja TimeLine: A Power Visualizer for Sensor Network Simulation. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, 2010.

[87] F. Österlind, N. Wirström, N. Tsiftes, N. Finne, T. Voigt, and A. Dunkels. StrawMAN: Making Sudden Traffic Surges Graceful in Low-Power Wireless Networks. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, Killarney, Ireland, June 2010.

[88] S. Park, A. Savvides, and M.B. Srivastava. SensorSim: a simulation framework for sensor networks. *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 104–111, 2000.

[89] B. Pásztor, M. Musolesi, and C. Mascolo. Opportunistic mobile sensor data collection with scar. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*, 2007.

[90] B. Pásztor, L. Mottola, C. Mascolo, G. Picco, S. Ellwood, and D. MacDonald. Selective reprogramming of mobile sensor networks through

social community detection. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Coimbra, Portugal, February 2010.

[91] N. Pereira, B. Andersson, and E. Tovar. WiDom: A dominance protocol for wireless medium access. *IEEE Transactions on Industrial Informatics*, 3(2):120, 2007.

[92] K. Pister and L. Doherty. TSMP: Time Synchronized Mesh Protocol. In *Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08)*, Orlando, Florida, USA, November 2008.

[93] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Baltimore, MD, USA, 2004. ISBN 1-58113-879-2.

[94] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, Los Angeles, CA, USA, April 2005.

[95] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale. PIP: A Connection-Oriented, Multi-Hop, Multi-Channel TDMA-based MAC for High Throughput Bulk Transfer. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zürich, Switzerland, 2010.

[96] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, pages 255–267, San Diego, California, USA, 2005.

[97] A. Reinhardt, P. Mogre, T. Koenig, and R. Steinmetz. SFHC.KOM: Stateful Header Compression for Wireless Sensor Networks. In *Proceedings of the 35th IEEE Conference on Local Computer Networks (LCN 2010)*, October 2010.

[98] O. Rensfelt, F. Hermans, C. Ferm, P. Gunningberg, and L. Larzon. Sensei-UU: A Nomadic Sensor Network Testbed Supporting Mobile Nodes. Technical report, 2010. Technical Report 2009-025, Department of Information Technology, Uppsala University.

[99] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-mac: a hybrid mac for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, San Diego, CA, USA, November 2005.

[100] M. Ringwald and K. Römer. BitMAC: A Deterministic, Collision-Free, and Robust MAC Protocol for Sensor Networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Istanbul, Turkey, January 2005.

[101] M. Ringwald and K. Römer. BurstMAC - An efficient MAC protocol for correlated traffic bursts. In *International Conference on Networked Sensing Systems (INSS)*, 2009.

[102] K. Römer and M. Ringwald. Increasing the visibility of sensor networks with passive distributed assertions. In *Proceedings of the 3rd ACM Workshop on Real-World Wireless Sensor Networks (REALWSN'08)*, Glasgow, United Kingdom, April 2008.

[103] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT : Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MobiHOC 2003)*, 2003.

[104] V. Shnayder, M. Hempstead, Chen B., G.W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, November 2004.

[105] V. Shnayder, M. Hempstead, B. Chen, and M. Welsh. PowerTOSSIM: Efficient Power Simulation for TinyOS Applications. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2004.

[106] Smart Meter Texas Team. Smart Meter Texas. Web page. URL `http://www.smartmetertexas.com`. Visited 2011-09-28.

[107] J. L. S. Sobrinho and A. S. Krishnakumar. Real-Time Traffic over the IEEE 802.11 Medium Access Control Layer. In *Bell Labs Technical Journal*, 1996.

[108] T. Sookoor, T. Hnat, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrodebugging: Global views of distributed program execution. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 141–154. ACM, 2009.

[109] K. Srinivasan, M. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: measuring wireless link burstiness. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, NC, USA, 2008.

[110] M. Strübe, R. Kapitza, K. Stengel, M. Daum, and F. Dressler. Stateful mobile modules for sensor networks. In *Proceedings of Distributed Computing in Sensor Systems (DCOSS)*, 2010.

[111] M. Strübe, S. Böhm, R. Kapitza, and F. Dressler. Realsim: Real-time mapping of real world sensor deployments into simulation scenarios. In *Proceedings of the 6th ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, 2011.

[112] Y. Sun, O. Gurewitz, and D. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, NC, USA, 2008.

[113] S. Sundaresan, I. Koren, Z. Koren, and C.M. Krishna. Event-driven adaptive duty-cycling in sensor networks. *International Journal of Sensor Networks*, 6(2):89–100, 2009.

[114] S. Sundresh, W. Kim, and G. Agha. SENS: A Sensor, Environment and Network Simulator. In *37th Annual Simulation Symposium (ANSS37)*, Arlington, VA, April 2004.

[115] M. Tancreti, M.S. Hossain, S. Bagchi, and V. Raghunathan. Aveksha: A Hardware-Software Approach for Non-intrusive Tracing and Profiling of Wireless Embedded Systems. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2011.

[116] J. Thelen, D. Goense, and K. Langendoen. Radio Wave Propagation in Potato Fields. In *1st Workshop on Wireless Network Measurements*, 2005.

[117] K. Thoelen, S. Michiels, and W. Joosen. Middleware for adaptive group communication in wireless sensor networks. In *Proceedings of the 2nd International ICST conference on Sensor Systems and Software*, 2010.

[118] K. Tindell, A. Burns, and A.J. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 1995.

[119] B.L. Titzer, D.K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, April 2005.

[120] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2005.

[121] UCB/LBNL/VINT. Network simulator - ns (version 2). Web page. URL `http://www.isi.edu/nsnam/ns/`. Visited 2011-08-28.

[122] J.P. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010. ISBN 978-0123751652.

[123] M. Wachs, J. Choi, J. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis. Visibility: A new metric for protocol design. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, 2007.

[124] D. Watson and M. Nesterenko. Mule: Hybrid Simulator for Testing and Debugging Wireless Sensor Networks. *Workshop on Sensor and Actor Network Protocols and Applications*, 2004.

[125] Y. Wen, W. Zhang, R. Wolski, and N. Chohan. Simulation-based augmented reality for sensor network development. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, 2007.

[126] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, Los Angeles, CA, USA, April 2005.

[127] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler. Marionette: Using RPC for Interactive Development and Debugging of Wireless Embedded Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, Nashville, TN, USA, 2006.

[128] G. Wittenburg and J. Schiller. Running realworld software on simulated wireless sensor nodes. In *Proc. of the ACM Workshop on Real-World Wireless Sensor Networks (ACM REALWSN'06)*, Uppsala, Sweden, June 2006.

[129] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networking (ACM MobiCom)*, Rome, Italy, 2001.

[130] J. Yang, M.L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: a comprehensive source-level debugger for wireless sensor networks. *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, pages 189–203, 2007.

[131] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, USA, June 2002.

[132] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Boulder, CO, USA, 2006.

[133] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.

[134] H. Zhang, A. Arora, Y. Choi, and M.G. Gouda. Reliable bursty convergecast in wireless sensor networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005.

[135] H. Zhang, F. Österlind, P. Soldati, T. Voigt, and M. Johansson. Rapid Convergecast on Commodity Hardware: Performance Limits and Optimal Policies. In *The IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON)*, Boston, Massachusetts, USA, June 2010.

[136] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 125–138, Boston, MA, USA, 2004. ACM. ISBN 1-58113-793-1.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 866

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology.

Distribution: publications.uu.se
urn:nbn:se:uu:diva-159886

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2011